Securing Sensitive Data with the Ingrian DataSecure Platform

Andrew Koyfman

Ingrian Networks, Redwood City, CA 94122, USA
akoyfman@ingrian.com
http://www.ingrian.com/

Abstract. Recent high profile data thefts have shown that perimeter defenses are not sufficient to secure important customer data. The damage caused by these thefts can be disastrous, and today an enterprise with poor data security may also find itself violating privacy legislation and be liable to civil lawsuits. The Ingrian DataSecure Platform presents an approach for protecting data inside the enterprise – and so to help eliminate many of the threats of data theft.

This paper demonstrates how an enterprise can prevent unauthorized data exposure by implementing column encryption in commercially available databases. Adding security at the database layer allows an enterprise to protect sensitive data without rewriting associated applications. Furthermore, large enterprises require scalable and easily administrable solutions. In order to satisfy this demand this paper introduces the concept of a Network-Attached Encryption Server, a central device with secure storage and extensive user access permissions for protecting persistent security credentials.

1 Introduction

Consumers and on-line shoppers have been educated to look for the little lock icon at the bottom of their browser to verify that they are connected securely to the website of their choice. The icon tells the consumer that he is using SSL and that the data in transit will be protected from eavesdroppers. While some potential weaknesses exist in the SSL protocol [1] the amount of time that is needed to capture the data and break the cryptography surrounding the communication far outweighs the sensitivity of the data that is being protected [2]. Attackers that want to acquire personal information have a far easier target – the database where these records are stored.

Getting access to the database allows attackers to gather millions of records of identifying information in one security breach. While enterprises worldwide spend $42 billion per year on IT security, [3] expensive breaches continue to occur. Attackers use a variety of ingenious methods to penetrate firewalls and other perimeter security defenses. Many attackers exploit unpatched systems or tunnel through existing applications and are thereby undetected by perimeter defense systems set up by the IT department. Further, most estimates cite that over 50% of the breaches are perpetrated by those inside the organization [4].

A.S. Patrick and M. Yung (Eds.): FC 2005, LNCS 3570, pp. 203 – 210, 2005.

In all these security breaches, after penetrating the perimeter defenses the attacker can access many corporate machines virtually unchallenged. If the database server is poorly protected, then the attacker will obtain information stored in customer records. Even access to the media containing raw database files is often sufficient for an attacker to gain the information he is seeking.

1.1 Damage Potential from Data Theft

An important question to ask is what damage can attackers cause by getting access to customer information, and how much effort should be spent on defending a system against these information gathering attacks? The answer, of course, depends on the nature of the company's business and the type of information stored. For a financial institution the consequences may be severe.

First, poor security may subject the company to a number of civil lawsuits. Consider the recent case of mass identity theft at BJ's Warehouse. After the theft, financial institutions had to reissue thousands of customers' credit cards as a precaution. For example, Sovereign Bank had to spend $1mil to reissue credit cards [5]. The affected companies are considering a civil lawsuit against BJ's to recoup their costs.
Second, the company's brand name may suffer as a result of disclosing the security breaches, especially if the company is a financial institution, or if the breaches are repetitive. Often disclosure is mandated by laws such as The California Privacy Law, SB-1386. Of course, the process of notification is expensive in itself.

Finally, the customer data itself is very valuable. It contains email addresses, demographics, and spending habits; information that is valuable to advertisers and competitors. Attackers can use this information for sending spam, as was the case in a data theft at Acxiom Corp [5], or they could disclose it to the company's competitors. The precise estimate of damage may be difficult to calculate, but the resulting loss of competitive advantage is obvious.

Each of these examples serves as a legitimate argument for protecting sensitive information such as customer records. Since perimeter security by itself is not enough to prevent data theft, the question arises of how best to limit the damage from possible security breaches.

2 Securing Customer Data in Databases

There are a number of approaches are available to organizations that can help them to protect customer data [6]. Perhaps the best and most secure solution is to write security-aware applications that encrypt and decrypt data prior to storing it and after retrieving it from back-end storage. However, it is expensive, if not impossible, to change existing, legacy applications to follow this model. Our system implements an alternative solution to perform encryption at the database level. The algorithm for enabling database encryption is well understood. We built tools that automate this process. Here, we will briefly mention the operations performed.

Consider a sample table containing the customer name, credit card number, order number, and the transaction date. (Fig. 1) In this table only the credit card number needs to be encrypted to prevent unauthorized access, prevent theft, and comply with privacy regulations.

NAME	CREDIT_CARD	ORDER_NUM	DATE
Jorge Chang	1234-5678-9012-2345	12345	8/25/04
Bruce Sandell	2234-5678-9012-2312	67890	2/29/04
... …	…	…	…

Fig. 1. A sample table CUSTOMER

Changing the database tables to encrypt data can be an error-prone and time consuming task if done by hand. The DataSecure Platform provides tools for DBAs to migrate existing data automatically. The tools follow a two step process, which is briefly discussed below. In Section 3, we will consider additional options for further restricting access to the encrypted data.

2.1 Creation of Encrypted Tables

During column encryption our tools first create a new table, CUSTOMER_ENC, and populate it with the data from the CUSTOMER table. Once the new table is created, it is desirable to transfer all data out of the CREDIT_CARD column in the CUSTOMER_ENC table to avoid any data type problems or other conversion issues. This is accomplished by creating a temporary table and populating it with the existing credit card column.

Next we adjust the data type in the CREDIT_CARD column to allow for storage of encrypted (binary) data, and to increase the size of the field if necessary. Finally, the CREDIT_CARD data is encrypted and re-imported into the table. (Fig. 2) The original table CUSTOMER is no longer needed and is deleted.

NAME	CREDIT_CARD	ORDER_NUM	DATE
Jorge Chang	23A2C3F243D52359F23 4BC67D2B57831	12345	8/25/04
Bruce Sandell	13B243F243D534A92F5 4A167C19578B3	67890	2/29/04
... …	…	…	…

Fig. 2. A sample table CUSTOMER_ENC

2.2 Creation of Triggers and Views

Of course, once the CUSTOMER table is deleted, all existing applications that were using this table will break. In order to allow the use of the current table, we create a

new view that has the same name as the original table, CUSTOMER. The new view has triggers associated with SELECT, INSERT, and UPDATE operations. These triggers execute stored procedures that encrypt and decrypt data as necessary. In this way, the existing application can still reference the same table name, and perform the same operations as before, while the data itself is now encrypted. (Fig. 3)

As one can see, the basic procedure for encrypting tables is fairly simple, even if some applications may require preparatory work to untangle complex inter-table relationships. A greater challenge lies in the proper storage and management of the secret keys used to encrypt the data.

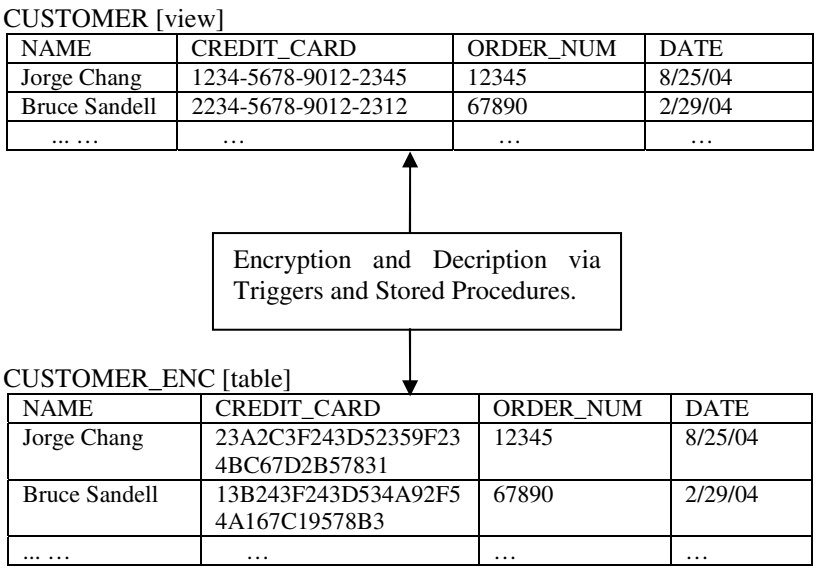

Fig. 3. Relationship between CUSTOMER view and CUSTOMER_ENC table

3 Key Management

The question of how to best store and distribute keys used for cryptography is a difficult one. This problem is particularly challenging in the case of database security, especially when the database runs on a cluster of machines. Usually, customers require that each machine in the cluster be able to perform the encryption operations on behalf of a user. Furthermore, only authorized users should have access to the encryption key. Finally, each machine should be able to perform cryptographic operations after an unattended restart, when the system administrator cannot type in the password.

3.1 Network-Attached Encryption (NAE) Device

The DataSecure Platform relies on a Network-Attached Encryption (NAE) server to handle all encryption operations associated with long-term data storage. (Fig. 4) The

NAE server can store all cryptographic keys used throughout the entire enterprise. Databases and other applications make calls to the NAE server and request it to perform an encryption or a decryption on their behalf. The NAE server verifies that the client has the permissions necessary to perform the operation and sends the result back to the client. The client only knows the key name, but not the actual key used for the operation. Thus, the key never leaves the NAE server and cannot be stored on insecure machines. This isolates all the keys' security to a single location which can be protected.

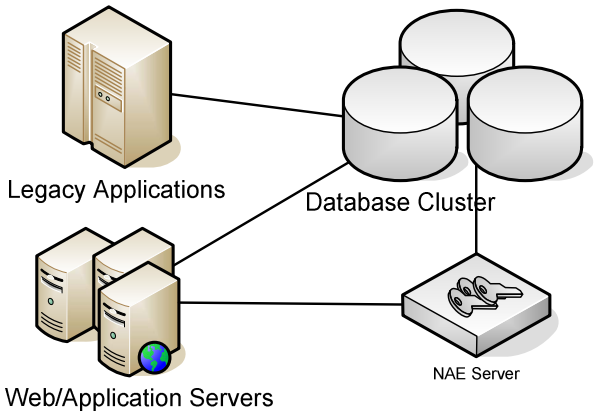

Fig. 4. NAE device in an enterprise setting. Legacy applications communicate with the database which in turn requests cryptographic operations from the NAE device. Newer web applications can access the database, or they can access the NAE device directly

3.2. User Access Permissions

In order to perform cryptographic operations, a client needs to authenticate itself to the NAE server. Using a username and password is still the most common way to perform this authentication. It is possible that some of these passwords are stored insecurely. For instance, developers may store them in plain-text in order to perform an unattended restart of the web server.

The NAE server can help to address this problem by restricting key use to authorized clients only. To accomplish this, the server maintains some meta-data along with the actual key material. The meta-data contains an extensive access permissions policy for potential key users. Some users may have full access to the key, while others may only be limited to either encryption or decryption operations. Additionally, the rate at which users can perform operations and the time of day when these operations may occur can be restricted in order to mitigate potential risks.

For example, consider an enterprise that has four different classes of users that need to access the CUSTOMER table from Figure 3. First a sample web application that needs to support unattended restarts (for example a checkout program), will populate the table with a new credit card number received from the customer. This

application would have the permissions to perform encryption operations, but would not be able to decrypt the CREDIT_CARD column. Therefore even if the password used by the checkout program was stolen, the thief would not be able to decrypt any of the data in the database.

A separate application, for instance one that performs batch credit card billing, will need to be able to decrypt the CREDIT_CARD column, but does not need to add new rows to the table. In order to reduce resource consumption during the day, batching operations are performed at night. Using the principle of least privilege, this application is given decrypt permissions only and is restricted to using its key between 2am and 4am.

On occasion users may call in with questions about their order. In order to help them, a customer service representative will need to access the CUSTOMER table. However, service representatives do not need to access the customer's credit card data. Therefore, they do not have permissions to perform encryption or decryption operations.

However, sometimes a customer is not satisfied with the order and demands a refund. In our example, it is the company's policy to have managers approve all refunds so, the customer service manager needs to have access to the customer's credit card information. The manager is human; hence it would be suspicious if he was issuing hundreds of refunds per second or working outside the regular business hours. Therefore, a service manager is restricted to making only one encryption or decryption per minute, and only during the day.

This separation of privileges limits the damage that may be caused by an attacker. The only way for an attacker to gain access to a large quantity of credit card numbers is to compromise the batch processing application. An enterprise would need to take sufficient precautions to make this infeasible.

Table 1. Key Policy Permissions

User	Can Encrypt	Can Decrypt	Rate	Time of Day
Checkout	y	n	Unl.	Unl.
Billing	n	y	Unl.	2AM – 4 AM
Service Rep.	n	n	NA	NA
Service Manager.	y	y	1/min	9AM - 5AM Mon. - Fri.

3.3 Disaster Recovery

Although not central to the security of the system, there is an additional benefit to centralizing key storage. Backup and disaster recovery is much easier when all of cryptographic information is stored in one place. An administrator no longer needs to worry about backing up different sets of keys that are scattered on machines throughout the enterprise. Instead, he only needs to back up the NAE server. Of course, these backup sets require additional protection to prevent them from falling into the wrong hands. The Ingrian NAE server encrypts the backups using a password provided by the administrator. Standard physical security practices should also be used to protect the back up media.

4 Performance

Improvements in security and manageability come at the expense of performance. Usually, turning on encryption dramatically increases the CPU usage on a machine. In some environments the NAE server model can ameliorate the CPU burden placed on the application or database server by offloading encryption operations. In these cases, the factors that limit performance are the network throughput and latency.

The initial design of the NAE server protocol followed the *init, update, final* model of encryption. The model requires the client to wait for a response from the server after each of the operations. While this works well for encryption of large chunks of data where network latency for the *init* and *final* calls is amortized over the entire operation, we have found that applications using the NAE server typically request encryption of short data chunks. In this case, the network latency has a severe impact on the number of operations a single thread can perform. We found that we can improve performance significantly by making a single *crypt* call to the server.

Table 2. Encryption and decryption operations performed per second against a single Ingrian NAE server using TCP to communicate. The server is a dual Penium 3, 1.26GHz cpu with 512Kb cache, 2Gb RAM and a Cavium crypto card (CN 1220-350). The client is a Pentium4HT, 3.0GHz cpu with 1Gb RAM. The AES algorithm uses a 256 bit key. The triple-Des algorithm uses a 168 bit key. The data size is 15 bytes, padded to 16 bytes, per operation

Encryption Method	Algorithm	1 thread (ops/sec)	10 threads (ops/sec)
init update final	AES/CBC/PKCS5	714	3,703
	3DES/CBC/PKCS5	666	3,703
single *crypt*	AES/CBC/PKCS5	1,250	9,090
	3DES/CBC/PKCS5	1,333	9,302
batch encryption	AES/CBC/PKCS5	31,648	NA
	3DES/CBC/PKCS5.	31,625	NA

The single crypt call performance is sufficient for most applications we have encountered. However, there is one class of applications, batch processing, which requires significantly higher throughput. For these applications we use a batch mode to communicate with the NAE server. The batch mode is optimized for a single thread to send multiple encryption requests without waiting to receive the answer from the server. This increases performance significantly.

The performance numbers in Table 2 were obtained by running a client test program against a single NAE server. NAE servers can be clustered to support a higher number of clients and operations.

5 Conclusion

Perimeter security and network security is not sufficient by itself to prevent attackers from stealing customer data records. In order to prevent expensive thefts, enterprises

must focus on protecting data in storage as well as data in transit. The Ingrian DataSecure platform combined with the NAE server provides a set of utilities that can aide administrators in securing enterprise data. The DBA is able to encrypt columns containing sensitive information and a Security Administrator can restrict access to the keys needed to retrieve the data. The NAE server also helps protects data throughout the enterprise by centralizing key storage, and by facilitating backup and recovery. While other methods for securing customer information exist, the benefit of using our approach is that it allows the enterprise to support existing, legacy applications without making expensive modifications.

References

1. Boneh, D., Brumley, D.: Remote Timing Attacks Are Practical, Proceedings of the 12th Usenix Security Symposium
2. Schneier, B.: CryptoGram Newsletter, March 15, 2003
3. Leyden, J.: Global IT Security Spending Hits $42bn, The Register, http://www.theregister.co.uk/2004/04/30/idc_security_booming/
4. Mogul, R.: Danger Within – Protecting your Company from Internal Security Attacks, Gartner, 2002
5. CBS News: "Big-Time ID Theft", Aug 13, 2004 http://www.cbsnews.com/stories/2004/08/13/tech/main635888.shtml
6. Mattsson, U.: A Database Encryption Solution That Is Protecting Against External And Internal Threats, And Meeting Regulatory Requirements, HelpNetSecurity, July 2004, http://www.net-security.org/article.php?id=715